# BAB II

# LANDASAN TEORI

## 2.1. Pengantar

Pada bab ini akan diuraikan hal-hal yang menjadi landasan teori dalam penelitian ini. Hal-hal yang akan diuraikan disini adalah teori tentang *Client-Server*, teknologi yang digunakan dalam perancangan aplikasi ini yaitu JAVA, J2ME, dan Jabber.

## 2.2. Teknologi Wireless Java

Secara konsep, teknologi wireless dapat dibagi dalam dua kategori, pertama untuk lokal dan kedua untuk area yang luas. Peralatan yang termasuk dalam kategori pertama misalnya adalah *remote control* untuk membuka atau mengunci mobil maupun garasi, telepon *cordless* 900Mhz, peralatan mainan dengan radio kontrol, atau jaringan *wireless*. Peralatan *wireless* jenis pertama ini hanya bekerja untuk daerah dengan jangkauan yang tidak terlalu jauh. Sedangkan peralatan jenis aplikasi yang kedua diantaranya adalah pager, ponsel, pda, dan sejenisnya. Jangkauan dari perangkat tersebut jauh lebih besar dari aplikasi jenis pertama. Karena jaringan yang ada di permukaan bumi berupa *cell-tower*, peralatan komunikasi bergerak seperti ponsel menerima layanan dari sebuah *wireless carrier* atau perusahaan yang mengoperasikan *celltower* tersebut. Aplikasi komunikasi bergerak, dalam perkembangan awal masing-masing *vendor* menghasilkan platform aplikasi dan sistem operasi sendiri. Sehingga sebuah peralatan ponsel Nokia dan Siemens mempunyai platform aplikasi masing-masing.

Perbedaaan aplikasi menyebabkan suatu platform aplikasi maupun sistem operasi dalam ponsel Nokia tidak dapat dijalankan dalam peralatan ponsel Siemens misalnya, sehingga berakibat memperburuk pengembangan aplikasi-aplikasi yang baru. Standardisasi yang dilakukan untuk membuat suatu bahasa pemrograman yang memiliki kebebasan platform atau *platform independence*. Salah satu teknologi Java adalah "*write once run everywhere*", sehingga portabilitas Java merupakan suatu kekuatan yang dimiliki Java. Java dijalankan pada sistem operasi apapun tanpa perlu kompilasi ulang program Java yang dibuat. Untuk komunikasi bergerak, *Sun Microsistem* mengenalkan Java 2 Micro Edition (J2ME) yang merupakan salah satu bagian teknologi Java yang digunakan untuk aplikasi Java yang berjalan pada perangkat *mobile device* dan teknologi aplikasi *wireless*.

### 2.2.1 Java Virtual Machine (JVM)

Java Virtual Machine adalah software yang berfungsi untuk menjalankan program Java supaya dapat dimengerti oleh komputer. Kode program Java ditulis menggunakan editor teks seperti Notepad, Textpad, Editplus, Jcreator dan lainnya. *Java Compiler* yang digunakan untuk mengkompilasi kode program Java dirancang untuk menghasilkan kode yang netral terhadap semua arsitektur perangkat keras (hardware) yang disebut sebagai Java Bytecode (*.class). Dan JVM merupakan basis dari Java platform dan menjembatani antara bytecode dengan hardware.

### 2.2.2 Java Application Programming Interface (Java API)

Java API merupakan komponen-komponen dan kelas Java yang sudah jadi, yang memiliki berbagai kemampuan. Kemampuan untuk menangani objek, string, angka dan sebagainya

1. *Applet*

   Java *Applet* merupakan program Java yang berjalan di atas *browser.* Penggunaan applet ini akan membuat halaman HTML lebih dinamis dan menarik.

2. *Java Networking*

   Sekumpulan API (*Application Programming Interface*) yang menyediakan fungsi-fungsi untuk aplikasi-aplikasi jaringan, seperti penyediaan akses untuk TCP, UDP, IP Adrress dan URL. Tetapi Java *Networking* tidak menyediakan akses untuk ICMP dikarenakan alasan keamanan dan pada kondisi umum hanya administrator (*root*) yang bisa memanfaatkan protokol ICMP.

3. *Java Database Connectivity* (JDBC)

   JDBC menyediakan sekumpulan API yang dapat digunakan untuk mengakses database seperti Oracle, MySQL, PostgreSQL, Microsoft SQL Server. Jadi keunggulan API JDBC dapat mengakses sumber data dan berjalan pada semua *platform* yang mempunyai *Java Viortual Machine* (JVM).

4. *Java Server Pages* (JSP)

   JSP adalah suatu teknologi web berbasis bahasa pemrograman Java dan berjalan pada *platform* Java. JSP merupakan pengembangan dari Servlet serta merupakan bagian dari teknologi Java 2 Platform, Enterprise Edition (J2EE).

5. *Java Card*

   Merupakan teknologi untuk membangun aplikasi pada sebuah *card electronic* seperti SIM Card pada ponsel.

### 2.2.3 Java 2 Platform

1. *Java 2 Standard Edition (J2SE)*

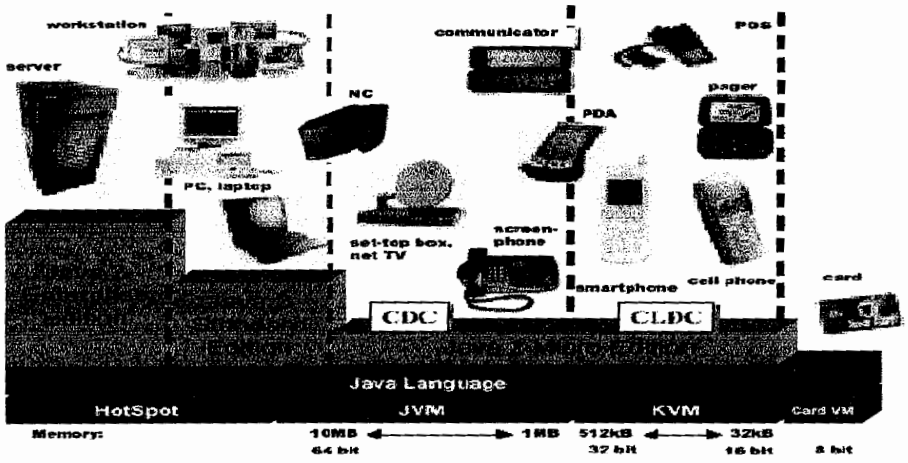
   Kategori ini digunakan untuk menjalankan dan mengembangkan aplikasi-aplikasi Java pada level PC (*Personal Computer*)

2. *Java 2 Enterprise Edition (J2EE)*

   Kategori ini digunakan untuk menjalankan dan mengembangkan aplikasi-aplikasi Java pada lingkungan *enterprise*, dengan menambah fungsionalitas Java semacam Java CORBA, *Servlet* dan Java XML *(Extensible Markup Language)*.

3. *Java 2 Micro Edition (J2ME)*

   Kategori ini digunakan untuk menjalankan dan mengembangkan aplikasi-aplikasi Java pada *handheld device* seperti ponsel, Palm, PDA dan Pocket PC.




Gambar 2.1. Lingkungan Kerja Teknologi Java

## 2.3. J2ME (*Java 2 Micro Edition*)

Seperti yang telah disebutkan di atas, J2ME dirancang untuk dapat menjalankan program Java pada perangkat-perangkat ponsel dan PDA, yang memiliki karakteristik berbeda dengan sebuah komputer biasa.

J2ME terdiri atas komponen-komponen :

a. *Java Virtual Machine* (JVM)
   Komponen ini untuk menjalankan program-program Java pada emulator.
b. Java API (*Application Programming Interface*)
   Merupakan kumpulan *library* untuk menjalankan dan mengembangkan program Java pada *handheld device*.
c. *Tools* untuk pengembangan aplikasi Java, yaitu emulator.

Dalam J2ME dibagi menjadi 2 bagian penting yaitu :

1. *Configuration*
   J2ME *Configuration* mendefinisikan lingkungan kerja J2ME *runtime*. J2ME *Configuration* menyediakan *library* standard yang mengimplementasikan fitur standard dari sebuah *handheld device*. Ada 2 J2ME *Configuration* saat ini yaitu :
   1. CLDC (*Connected Limited Device Configuration*)
      CLDC digunakan pada perangkat genggam yang memiliki ciri-ciri sebagai berikut :
      a. Memiliki memori sebesar 128 kilobyte untuk menjalankan dan imlementasikan aplikasi Java.
      b. Memiliki memori (minimum) sebesar 32 kilobytes untuk alokasi memori runtime

c. Memiliki koneksi jaringan walaupun terputus-putus dan tidak membutuhkan *bandwith* yang besar.
d. Daya listriknya kecil, biasanya menggunakan baterai.
e. Antarmuka pengguna terbatas.

2. CDC (Connected Device Configuration)

CDC digunakan pada perangkat genggam yang memiliki ciri-ciri sebagai berikut :

a. Memiliki memori (minimum) sebesar 512 kilobytes untuk menjalankan Java.
b. Memiliki memori (minimum) sebesar 256 kilobytes untuk alokasi memori runtime.
c. Memiliki koneksi jaringan yang terus menerus (tidak terputus-putus) dan bandwith yang besar.

Sederhananya, perbedaan antara CDC dan CLDC seperti dapat dilihat pada tabel 2.1 di bawah ini.

| **CLDC** | **CDC** |
|---|---|
| Mengimplementasikan subset dari J2SE | Mengimplementasikan seluruh fitur J2SE |
| JVM yang digunakan adalah KVM (*Kilobyte Virtual Machine*) | JVM yang digunakan adalah CVM (*Compact Virtual Machine*) |
| Digunakan pada perangkat *handheld* dengan ukuran memori 160-512 KB | Digunakan pada perangkat *handheld* dengan ukuran memori minimal 2 MB |
| Prosesor : 16 Bit-32 Bit | Prosesor : 32 Bit |

Tabel 2.1. Perbandingan antara CDC dan CLDC

2. *Profile*

Jika J2ME *Configuration* menyediakan *library* Java untuk implementasi fitur-fitur standard dari sebuah *handheld devices*, J2ME *Profile* menyediakan implementasi-implementasi tambahan yang sangat spesifik dari sebuah *handheld device*. Sebagai contoh ponsel memiliki kemampuan untuk menelpon ke suatu nomor ponsel lain karena ini merupakan kemampuan standard dari sebuah ponsel. Namun masing-masing jenis ponsel memiliki fitur sendiri-sendiri misalnya Siemens bisa menyimpan file MP3, Nokia memiliki aplikasi *game* tersendiri, dan lain sebagainya. Kemampuan standard itulah yang diimplementasikan J2ME *Configuration*, sedangkan kemampuan/fitur lain yang sangat bergantung pada jenis perangkat *handheld* yang digunakan akan diimplementasikan oleh J2ME *Profile*. Ada 5 kategori J2ME *Profile* saat ini yaitu :

a. MIDP(*Mobile Information Device Profile*)
b. FP(*Foundation Profile*)
c. *Personal Profile*
d. *RMI Profile*
e. *Personal Digital Assistance Profile*

MIDP menyediakan *library* Java untuk implementasi dasar antarmuka (GUI), implementasi jaringan (*networking*), database dan *timer*. MIDP dirancang khususnya untuk ponsel dan *pager*.

Seperti yang telah dijelaskan di atas, CLDC digunakan untuk implementasi program Java pada perangkat-perangkat keras dengan ukuran memori yang sangat terbatas, yakni pada kisaran 160

hingga 512 KB. Akibatnya fitur-fitur yang kurang penting untuk diimplementasikan dalam *handheld device* yang bersangkutan dari Java 2 harus dibuang (Ady Wicaksono, 2002). Fitur-fitur yang dibuang tersebut antara lain :

a. Tidak ada dukungan untuk *floating point*
   Kelas-kelas untuk perhitungan *floating point* yaitu java.lang.Float dan java.lang.Double tidak ada dalam CLDC.
b. Tidak ada dukungan untuk finalisasi objek
   *Garbage Collector* yang secara sederhananya digunakan untuk "bersih-bersih memori" membuang fungsi *finalize* pada kelas java.lang.Object sekalipun fungsi ini sangat penting di Java 2.
c. Penanganan kesalahan/*exception* yang terbatas
   CLDC hanya mendefinisikan 3 kelas berikut untuk penanganan kesalahan/*exception* :
   1. Kelas java.lang.Error
   2. Kelas java.lang.OutOfMemory
   3. Kelas java.lang.VirtualMachineError

Kelas-kelas tambahan yang tidak ada dalam J2SE adalah javax.microedition, yang menyediakan API(*Application Programming Interface*) untuk implementasi program pada *handheld device*. Dan mulai MIDP versi 2.0 juga disediakan Multimedia API(MMAPI) untuk implementasi audio dan video *player*.

## 2.4. Jabber

Jabber adalah sebuah protokol XML yang terbuka untuk pertukaran *message* dan *presence* yang *real-time* antara dua *user* di dalam jaringan Jabber. Jabber bukan sebuah perangkat lunak yang bersifat kaku, jadi memungkinkan pengembang-pengembang perangkat lunak untuk mengembangkannya. Aplikasi jabber yang utama adalah sebuah jaringan *instant messaging* yang menawarkan fungsionalitas yang sama dengan IM yang telah ada seperti AIM, ICQ, Yahoo! *Messenger*, MSN *Messenger* dll., namun teknologi Jabber menawarkan beberapa keuntungan :

a. *Open* (bersifat terbuka)

   Protokol jabber bersifat terbuka, gratis, publik, dan mudah dimengerti. Sekarang berbagai implementasi jabber terdiri dari aplikasi untuk *client*, *server*, *component*, dan *code library*.

b. *Standard*

   IETF menyusun inti dari protokol XML sebagai suatu teknologi IM dengan nama XMPP (*Extensible Messaging and Presence Protocol*), dan spesifikasi XMPP mengalami kemajuan cepat dibawah standard IETF.

c. *Proven* (terbukti handal)

   Jabber pertama kali dikembangkan oleh Jeremie Miller pada tahun 1998 dan tetap stabil sampai sekarang. Ratusan pengembang bekerja pada teknologi Jabber, sekarang sekitar sepuluh ribuan *server* Jabber berjalan di internet, dan ribuan orang menggunakan Jabber untuk IM. Pada

kenyataannya jumlah pengguna IM Jabber hampir menyamai jumlah pengguna ICQ.

d. *Decentralized* (terdesentralisasi)

Arsitektur jaringan Jabber sama dengan *email*, jadi setiap orang dapat menjalankan *server* Jabber mereka sendiri, memudahkan orang dan organisasi untuk mengontrol IM mereka.

e. *Secure* (aman)

Beberapa *server* Jabber terisolasi dari jaringan publik Jabber (contohnya pada jaringan intranet sebuah perusahan), dan sistem keamanan pada *server* XMPP menggunakan SASL (*Simple Authentication and Security Layer protocol*) dan TLS (*Transport Layer Security protocol*).

f. *Extensible*

Setiap orang dapat membangun fungsionalitas sendiri dan untuk mengatur interoperabilitas diatur oleh Jabber Software Foundation (JSF).

g. *Flexible* (Fleksibel)

Aplikasi Jabber selain IM meliputi manajemen jaringan, *content sindycation, collaboration tools, file sharing, gaming,* dan *remote sistems monitoring.*

h. *Diverse* (Diversifikasi)

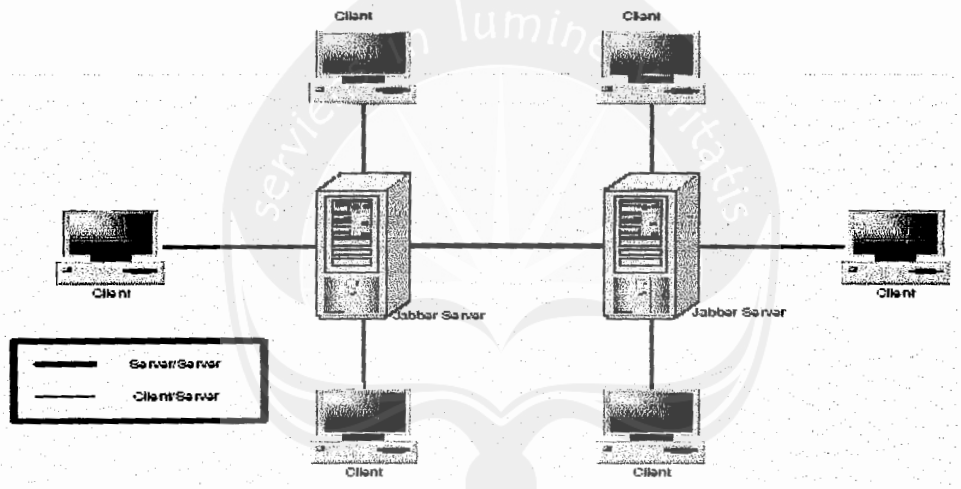
Berbagai perusahaan dan proyek *open-source* menggunakan protokol Jabber untuk membangun aplikasi dan layanan *real-time.*

Banyak kegunaan teknologi Jabber, pada awalnya teknologi Jabber bersifat *asynchronous, platform* IM yang dapat digunakan secara luas dan jaringan IM berdasarkan fungsinya hampir sama dengan sistem IM yang

resmi seperti *AOL Instant Messaging* (AIM) dan *Yahoo Instant Messaging*. Sebagai usaha menjadikan Jabber sebagai protokol standard *Instant Messaging*, pada Juni 2000 komunitas Jabber telah mempublikasikan protokol tersebut sebagai *Request for Comment* (RFC) pada *Internet Engineering Task Force* (IETF) sebagai bagian dari standard *Instant Messaging and Presence Protocol* (IMPP), tetapi IMPP ini tidak berjalan sukses. Pada bulan Mei 2001 *Jabber Community* dan Jabber Inc. membuat *Jabber Software Foundation* untuk menyediakan asisten organisasi secara langsung (*direct organizational assistance*) dan asisten teknis secara tidak langsung terhadap komunitas Jabber. Pada tahun 2002 *Internet Engineering Steering Group* (IESG) menyetujui formasi *Extensible Messaging and Presence Protocol Working Group* (XMPP) dengan *Internet Engineering Task Force* (IETF). Ruang lingkup *working group* adalah untuk mengeksplorasi dan dimana protokol tersebut digunakan, memodifikasi protokol yang sudah ada agar dapat memenuhi RFC 2799 seperti persyaratan yang ditentukan dalam spesifikasi *Common Presence and Instant Messaging* (CPIM). Fokus utama *working group* adalah membuat *XML stream* termasuk *stream* pada level *security* dan autentikasi, elemen data dan *namespace* yang dibutuhkan untuk mencapai dasar IM dan *Presence*. XMPP *working group* menerbitkan *XMPP Core Internet-Draft* sebagai dokumen yang menggambarkan fitur-fitur utama *Extensible Messaging* dan protokol *Presence*.

## 2.5. Tinjauan Teknologi Protokol Jabber

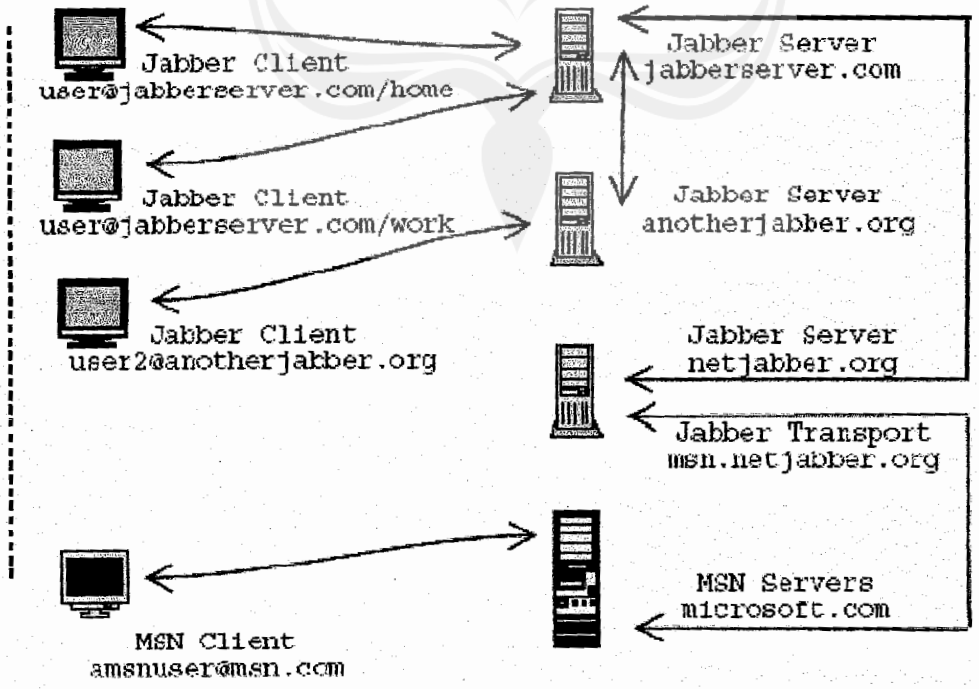
Jabber terkenal dengan arsitektur *client-server*nya, *client* Jabber dapat berkomunikasi dengan server Jabber pada *domain* Jabber mereka. *Domain* Jabber memiliki keuntungan yaitu kemampuannya dalam memisahkan zona komunikasi yang ditangani oleh *server* Jabber yang berbeda, tidak seperti kebanyakan sistem IM lainnya yang menggunakan satu server terpusat untuk seluruh zona komunikasi. Gambar 2.2 menunjukkan *stream* Jabber *client-server*.




Gambar 2.2 Aliran *client-server* Jabber

XMPP merupakan protokol hasil formalisasi IETF dari *streaming* protokol standar XML yang dikembangkan oleh *Jabber Community*. Protokol ini menghadirkan fitur lengkap untuk *Instant Messaging* dan *Presence* di atas *data transport layer* yang bersifat *dedicated*. Protokol ini telah stabil sejak tahun 1999. Jabber/XMPP adalah sebuah protokol yang telah didokumentasikan dengan baik dari seluruh protokol yang ada dan mudah untuk dipahami. Teknologi dasar dari XMPP menyangkut proses negosiasi XML *stream* antara *client* dan server, dengan

menggunakan *Simple Autentication and Security Layer* (SASL) dan *Transport Layer Security* (TLS) untuk mengamankan pengiriman datanya. Setelah melakukan autentikasi, selanjutnya pengguna dapat mengirimkan fragmen-fragmen XML sebagai hasil dari menjalankan fungsi-fungsi IM seperti mengirimkan pesan, chat dengan teman, merubah *status presence*, mengatur *contact list*, bergabung dengan *chatroom*, dan lain-lain. *Server* kemudian akan mengirimkan *message* kepada *server* lain melalui XML *stream* yang telah melalui proses negosiasi, berhubungan dengan syarat-syarat *security* untuk kemudian mencapai lokasi responden pengguna. XMPP kompatibel dengan teknologi Jabber yang sudah ada, sehingga menjamin interoperabilitas dengan jaringan yang ada saat ini. Aliran data pada protokol Jabber dapat dilihat pada Gambar 2.3




Gambar 2.3 Aliran data pada protokol Jabber

Cara Jabber/XMPP bekerja sering digambarkan seperti sebuah *router* XML artinya jika pesan dikirim dalam bentuk paket XML dan *route*-nya (pesan tersebut akan dikirim ke lokasi yang berdasar *content*-nya). Jabber di desain serupa dengan HTTP dan *e-mail* karena protokol ini relatif baru sampai saat ini Jabber memiliki sistem keamanan yang lebih baik. Jabber merupakan sistem jaringan terdistribusi yang menggunakan konektivitas *Domain Name Service* (DNS), Jabber mempunyai sebuah fasilitas *dial-back* yang tidak sama dengan *email* untuk menempatkan alamat, artinya seseorang yang melakukan *spamming* pada sebuah *server* dengan jumlah data yang besar secara cepat. *Password* dapat disimpan dan di autentikasi dengan berbagai cara termasuk menggunakan PGP/SSL. Saat ini tersedia banyak dokumentasi tentang komunikasi Jabber/XMPP dan protokol yang hanya sekali untuk didokumentasi secara keseluruhan. Jabber *support* terhadap sejumlah skema autentikasi dari algoritma *Hashing plaintext* dan standard SASL. Dengan menggunakan Jabber, komunikasi *client* ke *server* melalui SSL dan beberapa *client* menggunakan PGP berdasarkan *software* enkripsi. Sistem Jabber dapat juga terhubung ke sistem lainnya dengan sesuatu yang disebut *transport* yang berdasarkan *client emulation* dan dapat dijalankan pada *server* Jabber berdasarkan interoperabilitas antar protokol. Ditinjau dari sistem keamanan, pada protokol Jabber terjadi *client bugs* semacam *buffer overflow* yang berpengaruh pada versi khusus dari aplikasi yang secara langsung tidak dipengaruhi oleh virus atau *hacker*.

## 2.6. Arsitektur Jabber

### 2.6.1 Model *Client-Server*

Jabber menggunakan arsitektur *client-server*, bukan arsitektur langsung *peer-to-peer* seperti yang digunakan oleh sistem *messaging* lainnya. Akibatnya, seluruh data Jabber dikirim dari satu *client* ke *client* lainnya harus melewati minimal satu *server* Jabber. *Client* Jabber terhubung pada sebuah *server* Jabber pada TCP melalui port 5222. Koneksi ini selalu *on* untuk *session client* yang berjalan pada *server*, artinya *client* tidak dapat mengumpulkan pesan sebagai sebuah *email client*. Sebuah pesan diharapkan tersedia pada *client* dan dengan segera diharapkan *client messenger* sepanjang *client* masih terhubung. *Server* akan dapat menjajaki (*tracking*) apakah *client* masih *online* atau tidak, dan ketika *client* dalam kondisi *off-line* akan menyimpan beberapa pesan yang telah dikirim kepada *client* untuk menyediakan kapan dia akan terhubung lagi. Kekhasan yang dimiliki oleh protokol Jabber antara lain *modular server* dan *simple client* yang penjelasannya sebagai berikut :

**1. Modular server**

Server Jabber memiliki tiga peranan utama yaitu :

a. Menangani koneksi *client* dan berkomunikasi secara langsung dengan *client* Jabber
b. Berkomunikasi dengan server Jabber yang lain
c. Mengkoordinasikan beragam komponen *server* yang diasosiasikan dengan *server*

*Server* Jabber di desain modular dengan paket kode internal yang khusus sehingga dapat menangani fungsionalitasnya seperti registrasi, auntentikasi,

*present*, *contact list*, penyimpanan pesan yang berstatus *off-line* dan sebagainya. Selain itu *server* Jabber dapat dikembangkan dengan komponen eksternal yang memungkinkan *administrator server* untuk mensuplemen *server* pusat dengan layanan tambahan semacam gerbang untuk sistem *messaging* lainnya.

**2. Simple client**

Satu kriteria desain sistem Jabber bahwa ia harus memiliki kemampuan untuk mendukung *client* yang sederhana misalnya koneksi telnet pada *port* yang benar. Dalam hal ini tentu saja arsitektur Jabber memberikan sedikit batasan pada client. *Task-task* pada *client* Jabber harus dapat mengenal dan melengkapi :

a. Komunikasi dengan *server* Jabber melalui soket TCP
b. Melakukan *parsing* dan interpretasi XML dengan format yang baik melalui XML *stream*
c. Memahami tipe data utama Jabber (*message*, *presence* dan *iq*)

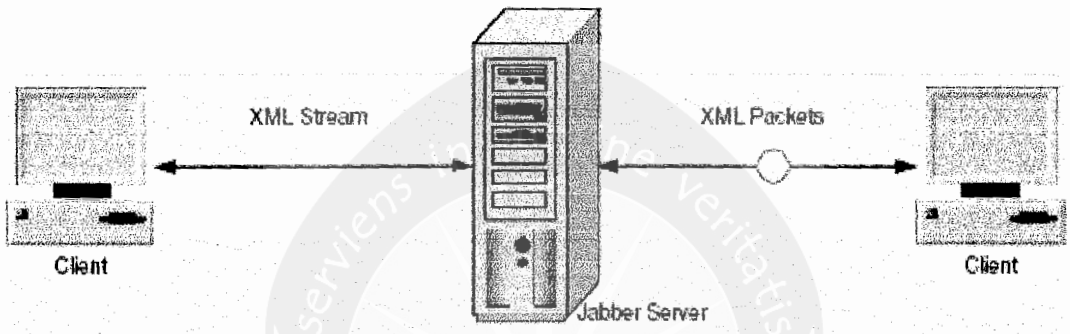
Keuntungan di dalam Jabber adalah dapat memindahkan kompleksitas dari *client* ke *server*. Secara praktis, banyak fungsi yang *low-level* pada *client* seperti proses *parsing* XML dan memahami tipe *data core* Jabber yang ditangani oleh *library-library client Jabber*, memungkinkan *client developer* untuk fokus pada *user interface*.

### 2.6.2 Format Data XML

Format data XML adalah bagian integral arsitektur Jabber karena sepenuhnya penting sehingga arsitektur secara fundamental dapat dikembangkan dan mampu diekspresikan dengan bentuk data yang terstruktur.

Gambar 2.4 menunjukkan model *messaging* Jabber yang digabungkan dengan 4 elemen utama yaitu :

a. Paket XML memuat data yang di *marked-up*,
b. XML *stream* yang digunakan untuk transportasi paket XML
c. *Client* dan *server* Jabber yang dapat saling dipertukarkan.




Gambar 2.4 Model *messaging* Jabber

Dengan terhubungnya *client* pada *server*, berarti membuka satu jalur ke *XML stream* dari *client* ke *server*, dan *server* merespon dengan satu jalur *XML stream* dari *server* ke *client*. Selanjutnya masing-masing *session* melibatkan dua XML *stream*. Seluruh komunikasi antara *client* dan *server* terjadi pada *stream* ini, contohnya adalah sebagai berikut :

```
<message from='jolie@jabber.com/home'
to='aim@rhymbox.com/work'>
<body>Hallo apa kabar! <body>
</message>
```

Ketika banyak *snippet* Jabber yang hanya sangat sederhana, format XML Jabber dapat juga ditingkatkan melalui *namespace* XML yang telah diatur oleh *Jabber Software Foundation* dan *namespace* disesuaikan untuk

aplikasi yang khusus. Hal ini yang membuat Jabber menjadi *platform* yang *powerful* untuk memilih struktur data yang digunakan termasuk *XML Remote Procedure Calls* (XML-RPC), *Resource Description Framework Site Summary* (RSFSS) dan *Scalable Vector Graphics* (SVG).

### 2.6.3 Jaringan Terdistribusi

Jaringan terdistribusi dalam hal ini bagaimana sebuah *server* Jabber dapat berkomunikasi dengan *server* Jabber lainnya dan dapat diakses melalui internet. Masing-masing *user* terhubung pada *home server* yang menerima informasi untuk mereka, selanjutnya *server* akan mentransfer data untuk kepemilikan *user*. Maka suatu domain dapat jalan pada *server* Jabber. Masing-masing fungsi *server* bebas terhadap yang lainnya, dan dikelola sendiri di dalam daftar *user*. *User* khusus diasosiasikan dengan *server* yang khusus pula dan alamat Jabber memiliki bentuk yang sama dengan alamat *email*. Jaringan yang terdistribusi ini menghasilkan sesuatu yang fleksibel, yaitu jaringan yang mampu terkontrol pada server yang memiliki skala yang lebih tinggi dibandingkan monolitik tetapi dengan syarat bahwa layanan yang terpusat hanya dapat jalan pada vendor IM yang resmi.

### 2.6.4 Standar Berdasarkan Pengalamatan

Ada beberapa perbedaan entitas pada Jabber untuk dapat berkomunikasi dengan yang lainnya. Entitas ini dapat direpresentasikan berupa *transport*, *groupchat room* atau *single Jabber user*. *Jabber ID* telah digunakan secara internal dan eksternal untuk menyatakan kepemilikan

atau *routing* informasi. Masing-masing *Jabber ID* memuat sekelompok order elemen. *JID* dibentuk dalam format [*node@domain/resource*], contohnya jolie@jabber.com/home

## 2.7. Komponen Utama Protokol Jabber

Ada tiga komponen utama pada protokol Jabber yang diandalkan dengan mekanisme *messaging* :

### 2.7.1 Message

Protokol *message* pada kenyataannya adalah protokol yang paling sederhana dalam Jabber. Banyak *traffic* di dalam jaringan Jabber yang termasuk dalam protokol *message*. Message terdiri dari 4 attribut, dan *zero* atau beberapa *child element*. Attribute dalam *message* adalah :

a. ***to*** : jenis yang diharapkan oleh pihak penerima pesan
b. ***from*** : jenis pesan yang dikirim
c. ***id*** : sebuah identifier unik yang bersifat opsional dengan tujuan dapat menjejaki *message*
d. ***type*** : sebuah spesifikasi opsional dari konteks percakapan sebuah *message*

Sedangkan atribut pada elemen anak antara lain :

a. ***body*** : isi tekstual dari *message*, secara normal termasuk tetapi tidak dibutuhkan.
b. ***subject*** : subjek dari *message*
c. ***thread*** : string acak yang di-*generated* oleh pengirim, digunakan untuk *tracking* sebuah *thread conversation*.
d. ***error*** : deskripsi pesan kesalahan

Versi protokol Jabber XMPP saat ini menggunakan standard yang merepresentasikan seluruh atribut dan

elemen anak yang ada pada *message protokol*. Contoh paket *message* ditunjukkan seperti berikut :

```
<message to='romeo@montague.net'
from='Juliet@capulet.com/balcony'
type='chat'>
<subject xml : lang='en'>
Apa kabar!
</subject>
<body xml : lang='en'>
hallo!!
</body>
<thread>e0f92794b9683a38</thread>
</message>
```

### 2.7.2 Presence

Protokol ini bertanggung jawab terhadap *subscription*, persetujuan, dan *update* informasi *presence* dalam komunitas Jabber. Atribut diasosiasikan dengan dengan protokol ini sama seperti pada protokol *message*, *presence* memiliki memiliki tipe atribut yang memiliki *state* sebagai berikut:

1. ***unavailable*** : *client* tidak lama tersedia untuk berkomunikasi
2. ***subscribe*** : pengirim mengirimkan *request* untuk *subscribe* terhadap *presence* penerima
3. ***subscribed*** : pengirim yang telah diizinkan terhadap *recipient* untuk menerima *presence* mereka.
4. ***unsubscribe*** : *subscription request* yang telah ditolak atau *subscription* yang telah di *cancel* sebelumnya.
5. ***probe*** : *request* dari *client* yang *presence* saat ini

6. ***error*** : pesan kesalahan yang berlangsung berdasarkan pemrosesan atau menyediakan paket *presence* yang telah dikirim sebelumnya.

Paket *presence* akan bernilai 0 atau 1 untuk masing-masing elemen anak berikut :

a. ***show*** : menggambarkan status yang tersedia dari entities atau *resource* yang spesifik. show memiliki empat nilai yang digunakan :
   1. **away** : *temporarily away*
   2. **chat** : bebas untuk *chat*
   3. **xa** : *extended away*
   4. **dnd** : *do not disturb*

b. ***status*** : merupakan deskripsi bahasa natural yang bersifat opsional yang mendeskripsikan status yang tersedia.

c. ***priority*** : bilangan *integer* bukan negatif yang menampilkan level prioritas pada *resource* yang terkoneksi, dengan 0 sebagai prioritas terendah.

d. ***error*** : deskripsi pesan kesalahan Contoh paket *presence* adalah sebagai berikut :

```
<presence>
from='juliet@capulet.com/balcony'
to='romeo@montague.net/orchard'>
<show> away </show>
<status> be right back </status>
<priority> 0 </priority>
</presence>
```

Paket *presence* dapat memuat *namespace* yang sesuai dengan elemen anak yang tidak
menganggu struktur *namespace* dan tag yang tersedia.

### 2.7.3 Info/Query

Protokol IQ adalah protokol Jabber yang terakhir dan yang paling peduli dibandingkan *message* dan protokol *presence*. IQ adalah protokol *request-response* yang umum sehingga di desain secara mudah untuk dikembangkan seperti HTTP yang merupakan medium *request-respon*. *Content* data dari *request* dan *respon* yang ditentukan dengan deklarasi *namespace* elemen anak secara langsung dari elemen IQ. Atribut diasosiasikan dengan protokol IQ memiliki atribut yang sama dengan protokol *message* dan *presence* kecuali jika protokol tersebut memiliki tipe atribut yang berbeda. Tipe atribut pada protokol *info/query* memiliki 4 nilai yang dapat digunakan ;

1. ***get*** : informasi *request*
2. ***set*** : menyediakan data yang dibutuhkan
3. ***result*** : respon terhadap *get* dan *set request* yang sukses
4. ***error*** : kesalahan yang terjadi dalam pemrosesan dan layanan *get* dan *set request*

Berikut adalah contoh paket IQ yang memblok *incoming message* dari JID yang khusus:

```
<iq type='set' id='msg1'>
<query xmlns='jabber:iq:privacy'>
<list name='message-jid-example'>
<item type='jid' value='tybalt@capulet.com'
action='deny' order='3'>
<message/>
</item>
</list>
</query>
</iq>
```

Protokol IQ ini sangat penting jika kita ingin membangun *server* berdasarkan kebijakan keamanan sistem yang harus dipenuhi oleh *client*. Jika sistem keamanan

*client* telah terpenuhi maka harus mendukung pula terhadap sistem keamanan pada sisi *server*.